REMARKS

Applicant is in receipt of the Office Action mailed July 1, 2008. Claims 1, 28, 29, and 30 have been amended. Claims 1-25 and 28-30 are pending in the case. Reconsideration of the present case is earnestly requested in light of the following remarks.

Telephone Interview Summary

On August 12, 2008, a telephone interview was conducted between Examiner Kawsar and Mark Williams (Reg. #50,658), in which the intended scope of the invention was discussed and compared to the cited art. More specifically, Applicant clarified to the Examiner that once the program execution is set to the rollback state, the program execution resumes executing based on or from *the rollback state in preparation for performing a program exit procedure*. The Examiner indicated that this clarified his conception of the invention scope, and suggested that such clarifying language be added to the independent claims, and that this would distinguish over the current cited art, but that further searching would be required. The Examiner further suggested amending means plus function claim 29 to recite "a *computer* system…", and indicated that this would resolve the section 101 rejection. Applicant agreed to amend the claims accordingly in a subsequent Response filed with a Request for Continued Examination.

Allowed Subject Matter

Applicant appreciates the allowed subject matter of claims 9-13, but believes that the claims as currently amended are patentably distinct and non-obvious over the cited art, as explained below.

Section 101 Rejections

Claim 29 was rejected under 35 U.S.C. 101 as being directed to non-statutory subject matter. Claim 29 is a means plus function claim. As explained in the previous Response, "means plus function" claims are a statutory claim category; *see, e.g.,* MPEP sections 2105-2107, 2181. Applicant further directs the Examiner's attention to 35 U.S.C. Section 112, sixth paragraph, which clearly states that the "means" in means plus

function claims specifically refer to structural elements disclosed in the Specification that operate to perform the claimed function. Such means are clearly described in Applicant's Specification, see, e.g., Figures 1A-5 and corresponding text, where various hardware devices, e.g., computers, vision systems, etc., are disclosed for implementing embodiments of the claimed invention. Thus, Applicant submits that the invention represented in claim 29 is not, and cannot be, "software per se", as asserted by the Examiner, since structural elements *are* included in the claim, per the relevant statute. Moreover, as established in State Street Bank & Trust Company v. Signature Financial Group, Inc., 149 F.3d 1368 (Fed. Cir. 1998), business method and software inventions may certainly be statutory, as long as they involve a practical application and produce a useful, concrete and tangible result. Applicant respectfully notes that per claim 29, once the task specification has been automatically generated, the function node is operable to execute in accordance with the generated task specification to perform at least a portion of the task, which is certainly a practical application with a tangible concrete result. Applicant thus respectfully submits that the invention as represented in claim 29 meets statutory requirements.

In the telephone interview summarized above, the Examiner suggested amending claim 29 to recite "a *computer* system...", and indicated that this would resolve the section 101 rejection. While Applicant believes the claim as previously written is statutory, per the above reasoning, in order to further prosecution in the case, Applicant as amended claim 29 accordingly, and respectfully requests removal of the section 101 rejection of claim 29.

Section 103 Rejections

Claims 1-8, 14-20, and 22-30 were rejected under 35 U.S.C. 103(a) as being unpatentable over Borkenhagen et al. (US Patent No. 6,076,157, "Borkenhagen"), in view of Fuchs et al. (US Patent No. 5,530,802, "Fuchs"). Applicant respectfully traverses the rejection.

Amended claim 1 recites:

1. A computer-accessible memory medium that stores program instructions for performing time-bounded execution of a program, wherein the program instructions are executable by a processor to perform:

initiating a timed program execution process, wherein the timed program execution process is operable to execute a program in a time-bounded manner;

initiating a timeout process, wherein the timeout process is operable to preempt the execution process to interrupt execution of the program;

configuring a timeout event, wherein the timeout event is an event indicating a timeout condition for the program;

the timed program execution process performing a time-bounded execution of the program, comprising:

a) determining and storing a rollback state for the program;

b) if the timeout event has not occurred, executing the program, wherein, during said executing, if the timeout event occurs,

c) the timeout process setting the timed program execution process to the rollback state, and disabling the timeout event; and

d) the timed program execution process resuming executing the program based on the rollback state with a timeout condition in preparation to perform a program exit procedure;

e) performing the program exit procedure;

disabling the timeout event;

terminating the timeout process; and

terminating the timed program execution process.

As explained in the above-summarized telephone interview, in Applicant's claimed invention as represented in claim 1, a program is required to execute within a determined duration, but in case of a program time-out, it is desired that the program exit,

but not be left in an unstable state. Thus, a roll-back state is determined for the program, where it is known that in the roll-back state the program is stable. Then, during execution of the program, if the program times out, the program is reset to the roll-back state (thus, putting the program into a stable state), and allowed to exit gracefully with a time-out condition. Thus, while the program is terminated, this termination occurs such that the program exits gracefully, i.e., in a stable state.

Applicant respectfully notes that Borkenhagen is directed to forced hardware thread switching, which is very different from Applicant's invention as recited in claim 1. More specifically, Borkenhagen's system is directed to managing hardware threads in a pipelined process such that, if a first thread stalls, e.g., due to memory latency in a multilevel cache system, the CPU can switch threads to perform useful work while the first thread is incapacitated. Thus, for example, as Borkenhagen makes clear, in Borkenhagen's system, when a thread (which Applicant notes is *not* a program) times out, e.g., due to inactivity ("not performing useful processing") or high latency, a thread switch is performed, i.e., switching processing from one thread to another, but this is not performed with respect to a (time-bounded) *program*, and in fact, Borkenhagen nowhere discusses time-bounded *program execution* at all. Moreover, Borkenhagen's thread switching doesn't terminate execution of a program, but rather pauses any execution being performed by the thread, e.g., until the thread is capable of running again. In other words, Borkenhagen is directed to maintaining hardware multi-thread efficiency by allocating processor time away from threads that are stalled or otherwise operating with high latency, and has nothing whatsoever to do with ensuring that an executing program either executes within a specified time, or exits gracefully.

Fuchs is directed to automatically working around a failure due to a fault detected during program execution. In response to detecting the fault, the method of Fuchs resets to a rollback point, then iteratively attempts to bypass the fault by reordering inputs to the program, where if the first reordering doesn't avoid the fault, a second reordering is attempted, and so on.

In direct contrast to Borkenhagen and Fuchs, per claim 1, the timed program execution process controls the program execution, where the timed program execution process and the timeout process are separate and distinct, e.g., they may execute in

different threads and where the timeout process affects the program execution process (which isn't taught by Borkenhagen)–not just switching threads (stopping the process from running), but rather, it changes the program execution process, resetting it to a previous state, specifically, the rollback state, and exiting gracefully with a timeout condition. There are numerous benefits to this approach (which are not provided by the cited art), e.g., in addition to ensuring a graceful exit from the execution process, the rollback state may be specified to ensure a valid state for system resources, e.g., allocated memory, processor mode/state. Fuchs, which is directed to software failure recovery, not time-bounded program execution, also fails to teach or suggest this functionality, as does the combination of Borkenhagen and Fuchs, as will now be explained.

Applicant recognizes that there are similarities between some terms in the cited references and Applicant's claim language, but does not believe that the combination of the cited references results in the specific functionality recited in Applicant's independent claims, summarized above. For example, Applicant notes that in Fuchs's fault bypass system and reorder recovery algorithm, the recovery process is performed in response to an application crashing or hanging, and results in restoring the application process to a checkpoint, reordering inputs logged since the checkpoint, and reprocessing the reordered inputs, i.e., continuing the application process by bypassing the fault, which Applicant believes is quite different from Applicant's graceful *program recovery and exit*. Nor does combining Fuchs's fault bypass system with Borkenhagen's hardware multithreading invention produce Applicant's claimed functionality, at least since, as noted above, Borkenhagen's system is directed to managing hardware threads in a pipelined process such that, if a first thread stalls, e.g., due to memory latency in a multilevel cache system, the CPU can switch threads to perform useful work while the first thread is incapacitated. This is quite different from the program-centric invention claimed by Applicant. Nowhere does Fuchs (or Borkenhagen) disclose rolling back to a rollback point, then exiting gracefully with a timeout condition, all in response to a *failure to execute a program within a specified time*.

Considering the claimed features more specifically, Applicant submits that there are numerous features and limitations of claim 1 that are not taught or suggested by the cited art.

For example, nowhere does the cited art disclose **e) performing the program exit procedure** (after rollback), as recited in claim 1.

Cited col.16:22-27 of Borkenhagen reads:

> If, however, the counter value is equal to the threshold value, no thread switch will occur until an instruction can execute, i.e., until forward progress occurs. Note that if the threshold register has value zero, at least one instruction must complete within the active thread before switching to another thread.

As may be seen, this text makes no mention of performing a program exit procedure, but rather describes conditions under which a thread switch may or may not occur. Nor does Fuchs disclose this feature. In fact, the only exiting Fuchs discloses is that of a "progressive retry recovery algorithm", which is the process that recovers the application, not the application that is rolled back.

Thus, the cited art fails to teach or suggest this claimed feature.

Additionally, nowhere does the cited art disclose: in response to a timeout event during execution, **c) the timeout process setting the timed program execution process to the rollback state, and disabling the timeout event; and d) the timed program execution process resuming executing the program based on the rollback state with a timeout condition in preparation to perform a program exit procedure**, as recited in claim 1.

The Office Action admits that Borkenhagen fails to disclose a) determining and storing a rollback state for the program, and (in response to a timeout event during execution) c) the timeout process setting the timed program execution process to the rollback state, and disabling the timeout event; and d) the timed program execution process resuming executing the program based on the rollback state with a timeout condition, but asserts that Fuchs remedies this admitted deficiency of Borkenhagen, citing col.2:51-60, col.3:24-30, and col.3:30-34. Applicant respectfully disagrees.

For example, cited col.3:24-30 reads:

> In one embodiment, a reorder recovery algorithm is provided that consists only of the receiver reorder step of the overall progressive

> retry algorithm. When information is known about the particular application process or the cause of the detected fault, it may be determined that the reorder recovery algorithm may be more appropriate for bypassing the fault.

As may be seen, this text in no way teaches or suggests "c) the timeout process setting the timed program execution process to the rollback state, and disabling the timeout event", contrary to the Office Action's assertion. Rather this citation simply discloses provision of a reorder recovery algorithm consisting only of a receiver reorder step in an overall progressive retry algorithm, whose use may be appropriate for bypassing faults in an application process. As Fuchs states in the Abstract, the reorder recovery algorithm attempts "to bypass the detected fault by reordering and reprocessing the inputs that have been received by the faulty application process", and has nothing to do with timed program execution. More specifically, this citation makes no mention of a timeout process setting a timed program execution process to a rollback state (in response to a timeout event), nor disabling the timeout event.

Applicant further notes that Fuchs in general fails to even mention a timeout process at all, nor a timeout event, and more specifically fails to disclose a timeout process rolling back a timed program execution process to a rollback state in response to a timeout event, and then disabling the timeout event.

Thus, the cited art fails to teach or suggest this claimed feature.

Cited col.3:30-34 reads:

> For example, the reorder recovery algorithm is particularly suitable for bypassing faults where the application process exhibits deterministic behavior, or where the detected fault results from the occurrence of a boundary condition or a racing condition.

Applicant notes that this text makes no mention of "**d) the timed program execution process resuming executing the program based on the rollback state with a timeout condition in preparation to perform a program exit procedure**", contrary to the Office Action's assertion. Rather this citation discusses under what conditions "the reorder recovery algorithm is particularly suitable for bypassing faults", e.g., "where the

application process exhibits deterministic behavior, or where the detected fault results from the occurrence of a boundary condition or a racing condition". As noted above, per Fuchs's Abstract, the reorder recovery algorithm attempts "to bypass the detected fault by reordering and reprocessing the inputs that have been received by the faulty application process", and has nothing to do with timed program execution. More specifically, this citation makes no mention of a timed program execution process *resuming executing a program based on a rollback state with a timeout condition* ***in preparation to perform a program exit procedure***. Moreover, as noted above, Fuchs nowhere even mention a timeout process at all, nor a timeout event, nor performing a program exit procedure. Nor does Fuchs nor Borkenhagen appear to even consider rolling back a program so that it can exit gracefully, and more specifically, in response to failing to execute within a specified time period. Neither Borkenhagen nor Fuchs is concerned with *timed execution of a program.*

As discussed above at length, the timed program execution process resuming executing the program based on the rollback state with a timeout condition allows the program to end gracefully, e.g., without leaving the system or its resources in unstable or inappropriate states, a functionality not provided by Borkenhagen and/or Fuchs.

Thus, the cited art fails to teach or suggest these claimed features.

Thus, for at least the reasons provided above, Applicant submits that Borkenhagen and Fuchs, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 1.

Applicant further submits that the Office Action has not provided a proper reason to combine Borkenhagen and Fuchs. For example, the only motivation to combine suggested by the Office Action is "because one of the [*sic*]ordinary skills of the art would want to be able to utilize the CPU performance by using the rollback state or any application that has failed or timeout for any event".

Applicant respectfully submits that wanting "to be able to utilize the CPU performance" has no bearing on the novel and beneficial functionality of Applicant's invention as represented by claim 1, at least for the reason that "utilizing the CPU performance" is not germane to resuming executing the program based on the rollback

state with a timeout condition (thereby exiting the program execution gracefully). In other words, a primary benefit of the functionality recited in claim 1 is that when a timeout occurs, and the program is rolled back and exits with a timeout condition, the system is prevented from ending up in an undesirable or unpredictable state, which is not really germane to improving the performance of a multi-threaded CPU. Thus, Applicant submits that Borkenhagen and Fuchs are not properly combinable to make a prima facie case of obviousness.

Moreover, even were Borkenhagen and Fuchs properly combinable, which Applicant argues they are not, the resulting combination would still not produce Applicant's invention as recited in claim 1, as discussed above at length.

Thus, for at least the reasons provided above, Applicant submits that claim 1, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claims 28, 29, and 30 include similar novel limitations as claim 1, and so the above arguments apply with equal force to these claims. Thus, for at least the reasons provided above, Applicant submits that claims 28, 29, and 30, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claim 21 was rejected under 35 U.S.C. 103(a) as being unpatentable over Borkenhagen in view of Fuchs, and further in view of Chamberlain (US Patent No. 6,438,749).

Applicant respectfully submits that since claim 21 depends from independent claim 1, which was shown above to be patentably distinct and non-obvious, and thus allowable, claim 21 is similarly patentably distinct and non-obvious, and thus allowable, for at least the reasons provided above.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

Removal of the section 103 rejection of claims 1-8, 14-21, and 22-30 is earnestly requested.

CONCLUSION

In light of the foregoing amendments and remarks, Applicant submits the application is now in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-81401/JCH.

Also filed herewith are the following items:

☒ Request for Continued Examination

☐ Other:

Respectfully submitted,

/Jeffrey C. Hood/

Jeffrey C. Hood, Reg. #35198
AGENT FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 2008-08-28 JCH/MSW